MapReduce and Relational Database Management Systems: competing or completing paradigms?

Dhouha Jemal
LARODEC / Tunis
dh.jemal@gmail.com

Rim Faiz
LARODEC / Tunis
rim.faiz@ihec.rnu.tn

Abstract

With the data volume that does not stop growing and the multitude of sources that led to diversity of structures, data processing needs are changing. Although, relational DBMSs remain the main data management technology for processing structured data, but faced with the massive growth in the volume of data, despite their evolution, relational databases, which have been proven for over 40 years, have reached their limits. Several organizations and researchers turned to MapReduce framework that has found great success in analyzing and processing large amounts of data on large clusters. In this paper, we will discuss MapReduce and Relational Database Management Systems as competing paradigms, and then as completing paradigms where we propose an integration approach to optimize OLAP queries process.

1 Introduction

The data is growing at an alarming speed in both volume and structure. The data explosion is not a new phenomenon; it is just accelerated in an incredible way and has an exponential number of technical and application challenges. Data generation is estimated of 2.5 trillion bytes of data every day. In addition, an IDC study predicts that overall data will grow by 50 times by 2020.

Data is the most precious asset of companies and can be mainspring of competitiveness and innovation. As presented in (Demirkan and Delen, 2013), many organizations noticed that the data they own and how they use it can make them different than others. That is why organizations need to be able to rapidly respond to market needs and changes, and it has become essential to have efficient and effective decision making processes with right data to make the decision the most adapted at a given moment. This explain the necessity to choose the right technology for processing and analyzing data.

As presented in (Ordonez, 2013), for more than forty, the relational database management systems have been the dominating technology to manage and query structured data.

However, the voluminous data which does not stop growing and the multitude of sources which led to diversity of structures challenge the needs of data processing. With these changes, the database world has been evolved and new models are presented. MapReduce is one such framework that met a big success for the applications that process large amounts of data. It is a powerful programing model characterized by its performance for heavy processing to be performed on a large volume of data that it can be a solution to have the best performance.

Several studies have been conducted to compare MapReduce and Relational DBMS. Some works present the MapReduce model as a replacement for the relational DBMSs due to its flexibility and performance, and other confirm the efficiency of the relational databases. In the other hand, many research works aim to use the two approaches together. In this environment of data explosion and diversity, the question that arises is what technology to use for data process and analysis for a particular application, how to benefit the data management systems diversity?

The aim of this work is to provide a broad comparison of the two technologies, presenting for each one its strengths and its weaknesses. Then, we propose an approach to integrate the two paradigms in order to optimize the online analytical processing (OLAP) queries process by minimizing the input/output cost in terms of the amount of data to manipulate, reading and writing throughout the query's execution process.

The remainder of this paper is organized as follows. In section 2, we introduce MapReduce. In section 3, we describe Relational DBMS. In section 4, we present our proposed integration approach for optimizing the online analytical processing (OLAP) queries Input/Output execution cost. Finally, section 5 concludes this paper and outlines our future work.

2 MapReduce

MapReduce is a programming model completed by Google, which was introduced by Dean and Ghemawat (2004). It was designed for processing large data sets with a parallel, distributed algorithm on a cluster.

MapReduce was created in order to simplify parallel processing and distributed data on a large number of machines with an abstraction that hides the details of the hardware layer to programmers: it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. Google uses the MapReduce model to deploy large variety of problems such as: generation of data for Google's production web search service, data mining, machine learning, etc.

The MapReduce programming model has been successfully used for many different purposes. These included: parallelizing the effort; distributing the data; handling node failures.

The term MapReduce actually refers to two separate and distinct tasks: Map and Reduce. The mapper is responsible for reading the data stored on disk and process them; it takes a set of data and converts it into another set of data: reads the input block and converts each record into a Key/Value pair. The reducer is responsible for consolidating the results from the map and then write them to disk; it takes the output from a map as input and combines those data tuples into a smaller set of tuples.

At first, Google developed their own DFS: the Google File System (GFS). As described in (McClean et al., 2013), MapReduce tasks run on top of Distributed File Systems (DFS). The distributed storage infrastructure store very large volumes of data on a large number of machines, and manipulate a distributed file system as if it were a single hard drive. The DFS deals with data in blocks. In order to prevent data loss, each block will be replicated across several machines to overcome a possible problem of a single machine failure. So, this model allows the user to focus on solving and implementing his problem.

Nevertheless, the lack is that the MapReduce is independent of the storage system, it can not take into account all the input data for an available index. This explains the critics mainly from the database community. As described in (Gruska and Martin, 2010), the database community sees the MapReduce as a step backwards from modern database systems, in view of the MapReduce is a very brute force approach and it lacks the optimizing and indexing capabilities of modern database systems.

MapReduce, the powerful tool characterized by its performance for heavy processing to be performed on a large volume of data that it can be a solution to have the best performance hence makes it very popular with companies that have large data processing centers such as Amazon and Facebook, and implemented in a number of places. However, Hadoop, the Apache Software Foundation open source and Java-based implementation of the MapReduce framework, has attracted the most interest. Firstly, this is due to the open source nature of the project, additionally to the strong support from Yahoo. Hadoop has its own extensible, and portable file system: Hadoop Distributed File System (HDFS) that provides high-throughput access to application data.

Since it is introduced by Google, a strong interest towards the MapReduce model is arising. Many research works aim to apply the ideas from multi-query optimization to optimize the processing of multiple jobs on the MapReduce paradigm by avoiding redundant computation in the MapReduce framework. In this direction, MRShare (Nykiel et al., 2010) has proposed two sharing techniques for a batch of jobs. The key idea behind this work is a grouping technique to merge multiple jobs that can benefit from the sharing opportunities into a single job. However, MRShare incurs a higher sorting cost compared to the naive technique . In (Wang and Chan, 2013) two new job sharing techniques are proposed: The generalize d grouping technique (GGT) that relaxes MRShare's requirement for sharing map output. The second technique is a materialization technique (MT) that partially materializes the map output of jobs in the map and reduce phase.

On the other hand, the Pig project at Yahoo (Olston et al., 2008), the SCOPE project at Microsoft (Chaiken et al., 2008), and the open source Hive project 2 introduce SQL-style declarative languages over the standard MapReduce model, aim to integrate declarative query constructs from the database community into MapReduce to allow greater data independence.

3 Relational DBMS

Since it was developed by Edgar Codd in 1970, as presented in (Shuxin and Indrakshi, 2005), the relational database (RDBMS) has been the dominant model for database management. RDBMS is the basis for SQL, and is a type of database management system (DBMS) that is based on the relational model which stores data in the form of related tables, and manages and queries structured data. Since the RDBMSs focus on extending the database system's capabilities and its processing abilities, RDBMSs have become a predominant powerful choice for the storage of information in new databases because they are easier to understand and use. What makes it powerful, is that it is based on relation between data; because the possibility of viewing the database in many different ways since the RDBMS require few assumptions about how data is related or how it will be extracted from the database. So, an important feature of relational systems is that a single database can be spread across several tables which might be related by common database table columns. RDBMS also provide relational operators to manipulate the data stored into the database tables. However, as discussed in (Hammes et al., 2014), the lack of the RDBMS model resides in the complexity and the time spent to design and normalize an efficient database. This is due to the several design steps and rules, which must be properly applied such as Primary Keys, Foreign Keys, Normal Forms, Data Types, etc. Relational Databases have about forty years of production experience, so the main strength to point out is the maturity of RDBMSs. That ensure that most trails have been explored and functionality optimized. For the user side, he must have the competence of a database designer to effectively normalize and organize the database, plus a database administrator to maintain the inevitable technical issues that will arise after deployment.

A lot of work has been done to compare the MapReduce model with parallel relational databases, such as (Pavlo et al., 2009), where experiments are conducted to compare Hadoop MapReduce with two parallel DBMSs in order to evaluate both parallel DBMS and the MapReduce model in terms of performance and development complexity. The study showed that both databases did not outperformed Hadoop for user-defined function. Many applications are difficult to express in SQL, hence the remedy of the user-defined function. Thus, the efficiency of the RDBMSs is in regular database tasks, but the user-defined function presents the main ability lack of this DBMS type.

A proof of improvement of the RDBMS model comes with the introduction of the Object-Oriented Database Relational Model (ORDBMS). It aims to utilize the benefits of object oriented theory in order to satisfy the need for a more programmatic flexibility. The basic goal presented in (Sabàu, 2007) for the Object-relational database is to bridge the gap between relational databases and the object-oriented modeling techniques used in programming languages. The most notable research project in this field is Postgres (Berkeley University, Californie); Illustra and PostgreSQL are the two products tracing this research.

4 MapReduce-RDBMS: integrating paradigms

In this section, the use of relational DBMS and MapReduce as complementary paradigms is considered.

It is important to pick the right database technology for the task at hand. Depending on what problem the organization is trying to solve, it will determine the technology that should be used.

Several comparative studies have been co ducted between MapReduce and parallel DBMS such as (Mchome, 2011) and (Pavlo et al., 2009). MapReduce has been presented as a replacement for the Parallel DBMS. While each system has its strengths and its weaknesses. However, an integration of the two systems is needed, and as proposed in (Stonebraker et al., 2010), MapReduce can be seen as a complement to a RDBMS for analytical applications, because different problems require complex analysis capabilities provided by both technologies.

In this context, we propose a model that integrates the MapReduce model and a relational DBMS PostgreSQL presented in (Worsley and Drake, 2002), in a goal of queries optimization. We suggest an OLAP queries process model in a goal of minimizing Input/Output costs in terms of the amount of data to manipulate, reading and writing throughout the execution process.

The basic idea behind our approach is based on the cost model to approve execution and selectivity of solutions based on the estimated cost of execution. To support the decision making process for analyzing data and extracting useful knowledge while minimizing costs, we propose to compare the estimates of the costs of running a query on Hadoop MapReduce compared to PostgreSQL to choose the least costly technology.

As the detailed analysis of the queries execution costs showed a gap mattering between both paradigms, hence the idea of the thorough analysis of the execution process of each query and the implied cost. So, to better control the cost difference between costs of Hadoop MapReduce versus PostgreSQL on each step of the query's execution process, we propose to dissect each query for a set of operations that demonstrates the process of executing the query and in order to control the different stages of the execution process of each query (decomposing the query to estimate the costs on each system for each individual operation). In this way, we can check the impact of the execution of each operation of a query on the overall cost and we can control the total cost of the query by controlling the partial cost of each operation in the information retrieval process. For this purpose, we suggest to provide a detailed execution plan for OLAP queries. This execution plan allows zooming on the sequence of steps of the process of executing a query. It details the various operations of the process highlighting the order of succession and dependence. In addition, it determines for each operation the amount of data involved and the dependence implemented in the succession of phases. These parameters will be needed to calculate the cost involved in each operation.

Having identified all operations performed during the query execution process the next step is then to calculate the cost implied in each operation independently, in both paradigms PostreSQL and MapReduce with the aim of controlling the estimated costs difference according to the operations as well as the total cost of query execution. At this stage we consider each operation independently to calculate an estimate of its cost execution on PostreSQL on one hand then on MapReduce on the other hand. For this goal, we propose to relay on a cost model for each system PostgreSQL and Hadoop MapReduce in order to estimate the I/O cost of each operation execution on both system independently.

We aim to estimate how expensive it is to preprocess each operation of each query on both systems. Therefore, controlling the cost implied by each operation as well as its influence on the total cost of the query, allows the control of the cost of each query to support the decision making process and the selectivity of the proposed solutions based on the criterion of cost minimization.

Based on a sample workload of OLAP queries, and having identifying the cost of each operation for each query executed on both systems independently, the results analysis can be useful to deduct a generalized smart model that integrates the two paradigms to process the OLAP queries in a cost minimization way.

5 Conclusion

Given the exploding data problem, the world of databases has evolved which aimed to escape the limitations of data processing and analysis. There has been a significant amount of work during the last two decades related to the needs of new supporting technologies for data processing and knowledge management, challenged by the rise of data generation and data structure diversity.

In this paper, we have investigated the MapReduce model in one hand, then the Relational DBMS technology in the other hand, in order to present strengths and weaknesses of each paradigm.

Although MapReduce was designed to cope with large amounts of unstructured data, there will be advantages in exploiting it in structured data processing. In this fashion, we have proposed in this paper a new OLAP queries process model integrating an RDBMS with the MapReduce framework in a goal of minimizing Input/Output costs in terms of the amount of data to manipulate, reading and writing throughout the execution process.

Combining MapReduce and RDBMS technologies has the potential to create very powerful systems. For this reason, we plan to investigate other types of integration for different applications.

Reference

Chaiken R., Jenkins B., Larson P.A., Ramsey B., Shakib D., Weaver S. and Zhou J. 2008. Scope: Easy and efficient parallel processing of massive data sets. *Proceedings of the VLDB Endowment*, volume 1 (2). 1265-1276.

Demirkan H. and Delen D. 2013. Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. *Decision Support Systems*, Volume 55 (1). 412-421.

Dean and Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation*, volume 6.

Gruska N. and Martin P. 2010. Integrating MapReduce and RDBMSs. *Proceedings of the 2010 Conference* *of the Center for Advanced Studies on Collaborative Research*, IBM Corp. 212-223.

Hammes D., Medero, H. and Mitchell H. 2014. Comparison of NoSQL and SQL Databases in the Cloud. *Southern Association for Information Systems (SAIS) Proceedings*. Paper 12.

McClean A., Conceicao RC., and O'Halloran M. 2013. A Comparison of MapReduce and Parallel Database Management Systems. *ICONS 2013, The Eighth International Conference on Systems*: 64-68.

Mchome M.L. 2011. Comparison study between MapReduce (MR) and parallel data management systems (DBMs) in large scale data anlysis. *Honors Projects Macalester College*.

Nykiel T., Potamias M., Mishra C., Kollios G. and Koudas N. 2010. Mrshare: sharing across multiple queries in mapreduce. *Proceedings of the VLDB Endowment* ,volume 3 (1-2). 494-505.

Ordonez C. 2013. Can we analyze big data inside a DBMS?. *Proceedings of the sixteenth international workshop on Data warehousing and OLAP*, ACM. 85-92.

Olston C., Reed B., Srivastava U., Kumar R. and Tomkins A. 2008. Pig latin: a not-so-foreign language for data processing. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data , ACM*. 1099-1110.

Pavlo A., Rasin A., Madden S., Stonebraker M., DeWitt D., Paulson E., Shrinivas L. and Abadi D.J. 2009. A comparison of approaches to large scale data analysis. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, ACM*. 165-178.

Shuxin Y. and Indrakshi R. 2005. Relational database operations modeling with UML. *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*. 927-932.

Sabàu G. 2007. Comparison of RDBMS, OODBMS and ORDBMS. *Informatica Economic*.

Stonebraker M., Abadi D., DeWitt D.J., Madden S., Paulson E., Pavlo A. and Rasin A. 2010. Mapreduce and parallel dbmss : friends or foes ?. *Communications of the ACM*, volume 53 (1). 64-71.

Wang G. and Chan CY. 2013. Multi-Query Optimization in MapReduce Framework. *Proceedings of the VLDB Endowment, 40th International Conference on Very Large Data Bases*, volume 7 (3).

Worsley J.C. and Drake J.D. 2002. Practical PostgreSQL. *O'Reilly and Associates Inc*